

Graph Model Selection via Random Walks*

Lin Li, William M. Campbell, Rajmonda S. Caceres

MIT Lincoln Laboratory

{lin.li, wcampbell, rajmonda.caceres}@ll.mit.edu

Abstract

In this paper, we present a novel approach based on the random walk process for finding meaningful representations of a graph model. Our approach leverages the transient behavior of many short random walks with novel initialization mechanisms to generate model discriminative features. These features are able to capture a more comprehensive structural signature of the underlying graph model. The resulting representation is invariant to both node permutation and the size of the graph, allowing direct comparison between large classes of graphs. We test our approach on two challenging model selection problems: the discrimination in the sparse regime of an Erdős-Renyi model from a stochastic block model and the planted clique problem. Our representation approach achieves performance that closely matches known theoretical limits in addition to being computationally simple and scalable to large graphs.

1 Introduction

Graphs are important data abstractions that allow us to analyze complex relational patterns in many application domains, such as social networks, information networks and protein networks. An active area of research focuses on theoretical models that define the generative mechanism of a graph. The mapping of an observed graph instance to a model allows us to apply the theoretical knowledge we have about the model and to make precise claims about the underlying structure of the data. Yet given the complexity and inherent noise in real datasets, it is still very challenging to identify the best model for a given observed graph. We discuss an approach for graph model selection that leverages embeddings of graphs in high dimensional feature space. In addition to gaining insight about graph structure, feature representations of graphs allow for the application of traditional tools in machine learning, many of which require continuous

vector representations as input.

In this paper, we introduce *Walk2Vec*, which uses random walks to characterize and compare structural properties of the underlying graphs. Our approach is based on the following intuition: structurally different graphs are likely to exhibit different random walk characteristics. For example, some graphs diffuse faster across nodes than others. Inspired by the intuition that diffusion properties reveal discriminating features for comparing graphs, we propose a flexible framework for mapping graphs or any substructures into an Euclidean feature space. Our approach has several important characteristics. It is invariant to both node labeling and the size of the graph, making it more appropriate for realistic applications. It leverages different diffusion instantiations on the same graph to capture a richer profile of graph structure. Finally, our approach maintains its performance robustness while being computationally efficient.

Various efforts in the literature [1, 2, 3, 4] have focused on learning vector representations of nodes or subgraphs in the graph. Our approach differs from the above approaches by offering a unifying and flexible framework that can learn a feature representation of a single node in the graph, a subgraph or the whole graph. At the same time, we learn representations that are both node label and graph size invariant. This is an important characteristic considering that in many practical settings, we need to compare graphs of different sizes or graphs with different node labelings. However, in many such settings differences in graph size and node labelings do not necessarily imply different graph structural properties. Finally, our approach follows a different and novel way of capturing the rich and diverse structural patterns in a graph. By using and combining different initialization mechanisms for random walks on the same graph, we generate a more comprehensive structural signature of the graph.

A much closely related line of work focuses on mapping the graph as a whole into a topological or spectral space, often in the context of the model selection or graph classification task. Many features are considered to represent the graph, from various density and path

*This work was sponsored by the Defense Advanced Research Projects Agency under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

features [5, 6], to distributions of frequent subgraphs or graphlets [7, 8], to spectral features of graph matrices [9, 10, 11]. Our approach leverages random walk based features to capture the essential structure of the graph. As we demonstrate in Sec. 5, for two extensively-studied model selection cases, our approach achieves classification performance very close to known theoretical results. Furthermore, when compared to topological embeddings of graphs [5, 6], we show that our random walk features lead to much more tightly clustered embeddings of similar graph instances.

Finally, in the graph kernel literature [12, 13, 14, 15, 16], similarity between two graphs is defined as a function of the number of matching random walks. A major focus in this literature is the reduction of computational complexity with various results giving polynomial time algorithms [12]. An important feature of our algorithm is its computational efficiency, making it scalable to big data settings.

2 Problem and General Approach

Our motivating problem is graph classification. Let $G = (V, E)$ denote a connected graph with node set V and edge set E . Let \mathcal{C}_1 and \mathcal{C}_2 be two distinct random graph models. The goal is to classify a graph G as being drawn from one of these distributions $G \sim \mathcal{C}_1$ or $G \sim \mathcal{C}_2$. Although the problem can be more general, in this paper, we focus on the problem of selecting the best model given a set of generative graph models.

Given an input graph G , our general approach is to use random walks with different initial distributions to map the graph G into a Euclidean space $\phi(G)$, and then use standard machine learning methods to perform supervised learning and assign the graph point $\phi(G)$ to its closest generative model.

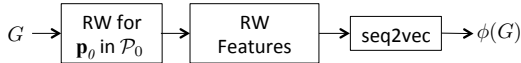


Figure 1: System for *Walk2Vec*

The proposed approach, *Walk2Vec*, for random walk features and mapping is shown in Figure 1. The first stage is to generate random walks on the graph G . Let \mathcal{P}_0 be a set of initial distributions over the nodes $\{1, \dots, n\}$ where $n = |V|$. For each distribution $\mathbf{p}_0 \in \mathcal{P}_0$, perform τ random walk steps on G and correlate the τ steps to produce a random walk feature vector $r(\mathbf{p}_0)$ (see Section 3.1). Then the resulting sequence of random walk features $\{r(\mathbf{p}_0)\}$ for all $\mathbf{p}_0 \in \mathcal{P}_0$ is then mapped into a single vector $\phi(G)$ in Euclidean space using the seq2vec operation as shown in Figure 1.

A desirable property for $\phi(G)$ is that it is invariant under graph isomorphism (permutation of the node

labels). In this paper, we achieve this property with two different strategies. The first one is via careful selection of the initial distributions in \mathcal{P}_0 to produce invariant random walk features $r(\mathbf{p}_0)$; each feature vector $r(\mathbf{p}_0)$ is independent of the node labels. In this case, the seq2vec can simply be a stacking operator and the resulting vector representation $\phi(G)$ of the graph G is also independent of the node labels and thus permutation invariant (see Section 3.2).

The second strategy is to generate random walk features that are localized to each of the nodes in the graph G . For example, the random walks are initialized from a single node. In this case, each feature vector $r(\mathbf{p}_0^i)$ is associated with a node label i where $i = 1, \dots, n$. To aggregate the sequence of vectors $\{r(\mathbf{p}_0^i)\}_{i=1}^n$, the seq2vec operation performs two steps: sparse coding and pooling. Sparse coding extracts high-level features from each $r(\mathbf{p}_0^i)$. The pooling step then combines these high-level features together and outputs a vector representation $\phi(G)$ of the graph G . The pooling operation has been used extensively in machine learning to achieve a better representation for classification. Additionally, in our case, pooling also provides a form of invariance under permutation of node labels (see Section 3.3).

Several generalizations of our problem and approach should be mentioned. First, although we consider graph model selection, the methods can be easily extended to graph classification or subgraph classification¹. Second, the mapping $\phi(\cdot)$ is robust and can be used for clustering, regression, etc.

3 Random walk features and mapping

Random walks have led to many important graph properties [17], i.e., hitting time, mixing time, commute time, etc., and can also provide us with considerable insight into the structure of the graph.

3.1 Random-Walk Graph Features The random walk process on a graph $G = (V, E)$ can be represented as follows. Let \mathbf{A} be the adjacency matrix associated with G : $A_{ij} = 1$ if nodes $(i, j) \in E$ and $A_{ij} = 0$ otherwise. The probability of moving from the current node to the neighbor is given by the transition matrix $\mathbf{W} = \mathbf{D}^{-1}\mathbf{A}$, where \mathbf{D} is the diagonal matrix of the degrees, i.e., $D_{ii} = d_i = \sum_j A_{ij}$. Suppose that the initial node is drawn from some initial probability distribution \mathbf{p}_0 . The probability distribution after t

¹Subgraph classification can be achieved by using a fixed local sampling method, e.g., the ego-net of a node or the subgraph induced by a breadth-first-search.

steps of random walk is given by

$$(3.1) \quad \mathbf{p}_t = \mathbf{W}^T \mathbf{p}_{t-1} = (\mathbf{W}^T)^t \mathbf{p}_0 .$$

For simplicity, we only consider the unweighted graphs. However, our approach can be easily extended to weighted graphs, where $A_{ij} \in \mathbb{R}_+$.

A fundamental property of a random walk on a connected, undirected and non-bipartite graph is that asymptotically, the probability distribution converges to a unique stationary distribution ω where $\omega_i = d_i / \sum_k d_k$. That is, asymptotically the probability of being at node i only depends on the degree of node i and not on the initial node.

Let us consider random walks on G with length τ . Note that: 1) the number of random walk steps τ must be sufficiently long to capture enough information about the graph; and 2) the probability distribution \mathbf{p}_t is biased towards the high-degree nodes as the number of random walk steps τ increases.

To generate the random walk feature on a graph, we now introduce a pair-wise distance matrix \mathbf{M} between probability distributions $\{\mathbf{p}_0, \mathbf{p}_1, \dots, \mathbf{p}_\tau\}$ up to step τ :

$$(3.2) \quad M_{st} = \|\mathbf{D}^{-\frac{1}{2}} \mathbf{p}_s - \mathbf{D}^{-\frac{1}{2}} \mathbf{p}_t\|_2 .$$

where $0 \leq s, t \leq \tau$. Elements in \mathbf{M} capture the temporal changes between various steps of the random walk. This distance can also be seen as the L^2 distance between the two probabilities \mathbf{p}_s and \mathbf{p}_t with respect to the stationary distribution ω [18]. This measure has also been used in [19] to compute distance between nodes using random walks on a graph for community detection and has shown good performance.

Given an initial distribution \mathbf{p}_0 , the random walk feature on graph G is defined by the function $r : \mathcal{P}_0 \rightarrow \mathbb{R}_+^d$ with $d = \frac{\tau^2 + \tau}{2}$ and

$$(3.3) \quad r(\mathbf{p}_0) = \text{triu}(\mathbf{M})$$

where \mathbf{M} is defined in Eqn. (3.2) and $\text{triu}(\cdot)$ returns the upper triangular elements of the matrix. Note that $r(\mathbf{p}_0)$ is independent of the graph size.

In this paper, we use the L^2 distance for computing the random walk feature. However, it is trivial to fit our framework to other distance metrics between two probabilities, such as total variation distance [20] and symmetric Kullback-Leibler (KL) divergence [21, 22]. One can also use the similarity matrix to generate the random walk feature. An example is to compute the following similarity matrix \mathbf{S} :

$$(3.4) \quad S_{st} = \frac{\mathbf{p}_s^T \mathbf{D}^{-1} \mathbf{p}_t}{\|\mathbf{D}^{-\frac{1}{2}} \mathbf{p}_s\|_2 \|\mathbf{D}^{-\frac{1}{2}} \mathbf{p}_t\|_2} .$$

where $0 \leq s, t \leq \tau$. and replace \mathbf{M} with \mathbf{S} in Eqn. (3.3).

3.2 Walk2Vec To map the input graph into a Euclidean space, we now restrict the random walk features $\{r(\mathbf{p}_0)\}$ to those that are invariant to node label. This property is addressed by choosing an appropriate initial probability distribution \mathbf{p}_0 for the random walks.

LEMMA 3.1. *Let \mathbf{A} be the adjacency matrix of the graph $G = (V, E)$ and let $\mathbf{p}_0 = \frac{\mathbf{g}(\mathbf{A})}{\|\mathbf{g}(\mathbf{A})\|_1}$ be an initial distribution on G , where \mathbf{g} is a nonnegative function of the adjacency matrix. If for any permutation matrix $\mathbf{\Pi}$ of compatible dimension,*

$$(3.5) \quad \mathbf{g}(\mathbf{\Pi A \Pi}^T) = \mathbf{\Pi g(A)} ,$$

then the distance matrix \mathbf{M} defined in Eqn. (3.3) is invariant under node permutation.

Proof. Suppose \mathbf{A} and $\tilde{\mathbf{A}}$ are two adjacency matrices, where there exists a permutation $\mathbf{\Pi}$ such that $\tilde{\mathbf{A}} = \mathbf{\Pi A \Pi}^T$, i.e., \mathbf{A} and $\tilde{\mathbf{A}}$ represent the same graph with a node permutation. Then their associated degree matrices and transition matrices follow the relation $\tilde{\mathbf{D}} = \mathbf{\Pi D \Pi}^T$ and $\tilde{\mathbf{W}} = \mathbf{\Pi W_1 \Pi}^T$, respectively. And the associated random walk distributions $\mathbf{p}_{t|\mathbf{A}}$ and $\mathbf{p}_{t|\tilde{\mathbf{A}}}$ at step t are

$$(3.6) \quad \mathbf{p}_{t|\tilde{\mathbf{A}}} = (\tilde{\mathbf{W}}^T)^t \mathbf{p}_{0|\tilde{\mathbf{A}}} = \mathbf{\Pi}(\mathbf{W}^T)^t \mathbf{\Pi}^T \mathbf{p}_{0|\tilde{\mathbf{A}}} ,$$

where $\mathbf{p}_{0|\mathbf{A}} := \frac{\mathbf{g}(\mathbf{A})}{\|\mathbf{g}(\mathbf{A})\|_1}$ and $\mathbf{p}_{0|\tilde{\mathbf{A}}} := \frac{\mathbf{g}(\tilde{\mathbf{A}})}{\|\mathbf{g}(\tilde{\mathbf{A}})\|_1}$. Then it follows from Eqn. (3.5) that $\mathbf{p}_{0|\tilde{\mathbf{A}}} = \mathbf{\Pi p}_{0|\mathbf{A}}$. Subsequently, Eqn. (3.6) becomes

$$(3.7) \quad \mathbf{p}_{t|\tilde{\mathbf{A}}} = \mathbf{\Pi}(\mathbf{W}^T)^t \mathbf{p}_{0|\mathbf{A}} = \mathbf{\Pi p}_{t|\mathbf{A}} .$$

Then it is easy to check that the distance matrix \mathbf{M} is invariant under the permutation $\mathbf{\Pi}$.

Examples of \mathbf{p}_0 that satisfy the condition in Lemma 3.1 include the uniform distribution, normalized centrality vector, normalized local clustering coefficient, etc. Additionally, $g(i)$ can also be a function that selects one (or a subset) of the nodes, i.e., the node with the highest centrality value or the highest clustering coefficient. Although there are many ways of selecting the initial distribution \mathbf{p}_0 that is permutation invariant. In Section 4, we will show one way of selecting the set of initial distributions that gives good results.

Suppose that \mathcal{P}_0 is the set of initial distributions over the graph; each $\mathbf{p}_0 \in \mathcal{P}_0$ leads to a permutation-invariant feature vector $\mathbf{x} = r(\mathbf{p}_0)$. Then a common approach to construct the mapping $\phi(G)$ is by stacking the sequence of random walk features $\{r(\mathbf{p}_0)\}$ into a single vector; see Section 4 for an example.

3.3 Walk2Vec-SC As discussed previously, one way to generate a mapping $\phi(G)$ of the graph G is to restrict each initial distribution $\mathbf{p}_0 \in \mathcal{P}_0$ such that the random walk feature $r(\mathbf{p}_0)$ is invariant to permutation of node labels. Alternatively, one can choose a set of initial distributions that are localized to the nodes of G . The permutation invariant property of the mapping $\phi(G)$ can be achieved by sparse coding (SC) the localized random walk features, followed by a pooling operation.

To extract localized random walk features, we use a set of initial probability distributions $\mathcal{P}_0 = \{\mathbf{p}_0^1, \dots, \mathbf{p}_0^n\}$; each \mathbf{p}_0^i is localized in the graph to node i . Several examples are: 1) a delta distribution which is one only on the i th node, $\mathbf{p}_0^i = \mathbf{e}_i$ where \mathbf{e}_i is the i th column of the identity matrix, or 2) a uniform distribution on the ego-net of the i th node. We then find random walk features as in (3.3) for each $\mathbf{p}_0^i \in \mathcal{P}_0$ to obtain a sequence of feature vectors $\mathbf{x}_i = r(\mathbf{p}_0^i)$.

If we combined the vectors \mathbf{x}_i using a simple function such as averaging $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, then the output would be a single vector that is also permutation invariant. However, significant information would be lost in the averaged vector $\bar{\mathbf{x}}$. A better way to approach this problem is to find a compact high-level representation for each of the feature vectors \mathbf{x}_i using sparse coding [23, 24] and then aggregate these high-level representations together using pooling.

We train a matrix \mathbf{D} , the dictionary, that is used to represent an input \mathbf{x}_i as $\mathbf{D}\mathbf{y}_i \approx \mathbf{x}_i$ where \mathbf{y}_i is a sparse vector. The dictionary is overcomplete; that is, redundant information is included to allow a sparse solution for \mathbf{y}_i . For a known \mathbf{D} , we use the LASSO criterion [25] for sparse coding

$$(3.8) \quad \mathbf{y}_i = \underset{\hat{\mathbf{y}}_i}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{D}\hat{\mathbf{y}}_i - \mathbf{x}_i\|_2^2 + \lambda_1 \|\hat{\mathbf{y}}_i\|_1.$$

The ℓ_1 penalty in (3.8) encourages sparsity in \mathbf{y}_i . Dictionary learning is via the methods in [26], and LASSO is solved with the LARS algorithm [27], both in the SPAMS software package. Intuitively, the dictionary atoms (columns of \mathbf{D}) represent different random walks, and the sparse coordinates \mathbf{y}_i are the atoms seen at node i .

The mapping $\phi(G)$ is then computed by pooling the sparse coded vectors \mathbf{y}_i across all i . Pooling is either average pooling $\phi(G) = \frac{1}{n} \sum_{i=1}^n \mathbf{y}_i$ or max pooling $\phi(G)_j = \operatorname{argmax}_i y_{i,j}$, where $y_{i,j}$ is the j th component of \mathbf{y}_i and $\phi(G)_j$ is the j th component of $\phi(G)$. Sparse coding and pooling perform the *seq2vec* operation in Figure 1. Pooling has the property that it creates a permutation invariant mapping of the graph. That is, reordering the node labels will not change the mapping $\phi(G)$.

4 Experiment Setup

For Walk2Vec, we first need to select a set of initial distributions that satisfy the condition in Lemma 3.1. In the experiment, we consider a set of four initial distributions $\mathcal{P}_0 = \{\mathbf{p}_0^{\max}, \mathbf{p}_0^{\min}, \mathbf{p}_0^{\text{median}}, \mathbf{p}_0^{\text{mean}}\}$. Specifically, \mathbf{p}_0^{\max} (or \mathbf{p}_0^{\min}) corresponds to a delta distribution \mathbf{e}_i , where i is the index of the node that has the maximum (or minimum) node degree in the graph. Similarly, $\mathbf{p}_0^{\text{median}}$ (or $\mathbf{p}_0^{\text{mean}}$) corresponds to a delta distribution \mathbf{e}_i , where i is the index of the node whose degree is the closest to the median (or mean) node degree in the graph. In other words, the random walks on G are initialized from each of the four above-mentioned type of nodes. Note that to satisfy the condition in Eqn. (3.5), the selected nodes also need to be unique. In the case that there exist more than one maximum (or minimum) degree node, we pick the one that has the maximum (or minimum) PageRank in the graph. A similar strategy is used in selecting the median and mean degree node in the graph. The resulting representation of the graph is given by stacking all the feature vectors $\phi(G) = [r(\mathbf{p}_0^{\max})^T, r(\mathbf{p}_0^{\min})^T, r(\mathbf{p}_0^{\text{median}})^T, r(\mathbf{p}_0^{\text{mean}})^T]^T$.

The setup for the Walk2Vec-SC system is as follows. For each graph, we use the set $\mathcal{P}_0 = \{\mathbf{p}_0^1, \dots, \mathbf{p}_0^n\}$, where $\mathbf{p}_0^i = \mathbf{e}_i$, to initiate the random walk process. This leads to a sequence of feature vectors $\mathbf{x}_i = r(\mathbf{p}_0^i)$. Given a training set, a dictionary of 100 atoms is trained using the SPAMS tool for Python. The dictionary is used to convert the random features into sparse vectors for each node using (3.8), and we set $\lambda_1 = 0.15$ for all the experiments. After computing sparse vectors, for each graph the mapping $\phi(G)$ is found using pooling.

Furthermore, we train a random forest classifier [28] and use the learned model to classify a collection of unlabeled graph instances. All random forest classifiers are trained with 100 decision trees.

5 Graph Model Selection: Case Studies

To validate our new approach, we apply Walk2Vec and Walk2Vec-SC to two graph model selection problems: 1) Erdős-Renyi vs. stochastic block model, and 2) planted clique problem.

5.1 Erdős-Renyi vs. Stochastic Block Model

The problem of distinguishing between an Erdős-Renyi (ER) graph [29] and a stochastic block model (SBM) [30] is as follows. Let p be the connection probability for the ER graph. Consider the case of a SBM graph with two communities (blocks) of equal size $n/2$ where $n = |V|$. The cross-community probability is p_{out} and the within-community probability is p_{in} ; the density is given by $(p_{\text{in}} + p_{\text{out}})/2$. As the difference $p_{\text{in}} - p_{\text{out}} > 0$ becomes smaller, the SBM graphs be-

come harder to distinguish from ER graphs of the same density. Let $\delta = (p_{\text{in}} - p_{\text{out}})/n$. For graphs of the same density, i.e., $p = (p_{\text{in}} + p_{\text{out}})/2$, the theoretical limit [31] for discriminating the two models is

$$(5.9) \quad \delta_{\text{crit.}} = 2\sqrt{\frac{p}{n}}.$$

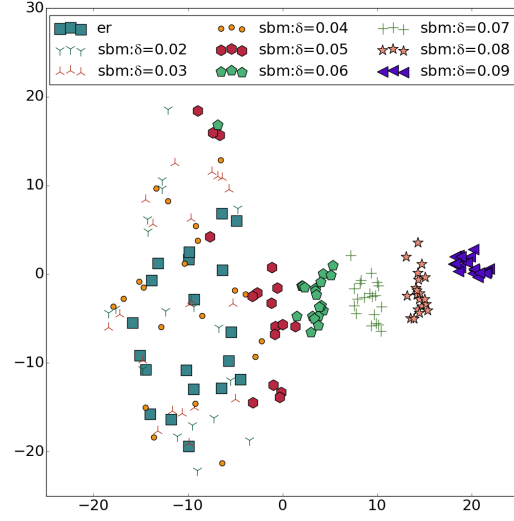
This limit offers a precise mechanism for quantifying the robustness of our two graph mapping algorithms.

For each pair of parameters (p, δ) , we generate 1000 ER graphs with density p and 1000 SBM graphs with p_{in} and p_{out} such that $(p_{\text{in}} - p_{\text{out}})/n = \delta$ and the density $(p_{\text{in}} + p_{\text{out}})/2 = p$. All graphs are generated with $n = 1000$ number of nodes.

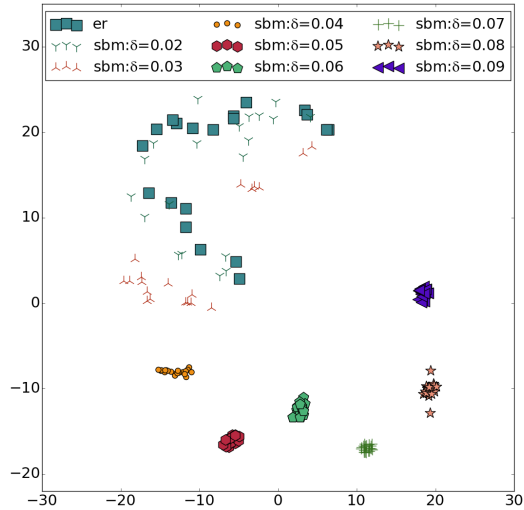
Figure 2 shows the 2-dimensional embeddings of vector representations $\phi(G)$ of ER graphs with $p = 0.05$ and SBM graphs with various δ values. The low-dimensional embedding is performed using the Principal Component Analysis (PCA). Specifically, Figure 2(a) shows the 2-dimensional embeddings of the Walk2Vec graph representations. Observe that for large δ value, the SBM graphs are far away from the ER graphs. As the δ value decreases, the SBM graphs get closer to the ER graphs. Figure 2(b) shows the 2-dimensional embeddings of the Walk2Vec-SC graph representations. Observe that for large δ value, the SBM graphs form clusters. One can almost see a loop as the δ value decreases. This implies that the Walk2Vec-SC method could also be useful in estimating the SBM parameter δ .

Finally, observe how, especially for the Walk2Vec-SC embeddings, stochastic block instances of the same δ parameter cluster together in space. This is a very desired property of graph embeddings since it demonstrates robustness and stability of such embeddings in the presence of model randomness or noise. Furthermore, in a similar problem setting, where the goal was to discriminate between graph instances generated by two different model parameters, we would prefer the separation of similar instances into tightly-knit, separable clusters.

After mapping each graph into its vector representation $\phi(G)$ using the proposed methods, we then train a random forest classifier on the 500 vector instances of each model and tested on the 500 vector instances of each model. Fig. 3 shows the AUC performance on the ER vs. SBM problem using Walk2Vec as δ increases and for various number of random walk steps. Here $p = 0.05$ and $n = 1000$. The dashed vertical line represent the theoretical threshold $\delta_{\text{crit.}}$ for discriminating ER and SBM models. Observe that the phase transition curve gets sharper as the number of random walk steps τ increases. For $\tau > 10$, increasing τ has little effect on the performance. For the rest of the experiment, we set



(a) Walk2Vec



(b) Walk2Vec-SC

Figure 2: Two-dimensional embeddings of the vector graph representations of ER graphs and SBM graphs. All vector representations are computed using $\tau = 15$ random walk steps.

$\tau = 15$.

The two heat maps in Fig. 4 show the results of the graph classification using Walk2Vec and Walk2Vec-SC, respectively, for various densities p and SBM δ values. The dark red corresponds to $AUC \approx 1$ and dark blue represent $AUC \approx 0.5$ (i.e., random detection). The dashed line represents the theoretical limit $\delta_{\text{crit.}}$. Observe that simulations agree well with the analytical prediction. In particular, the Walk2Vec-SC system

exhibits a very sharp phase transition that almost overlaps with the analytical prediction at all densities p .

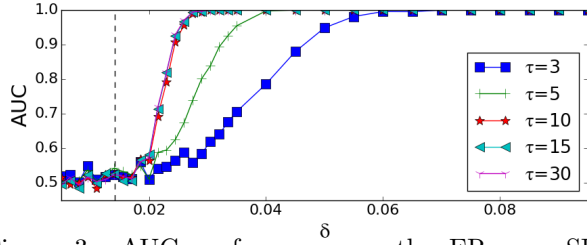


Figure 3: AUC performance on the ER vs. SBM problem using Walk2Vec and with varying random walk steps. The dashed line corresponds to the analytical prediction of the phase transition $\delta_{\text{crit.}}$.

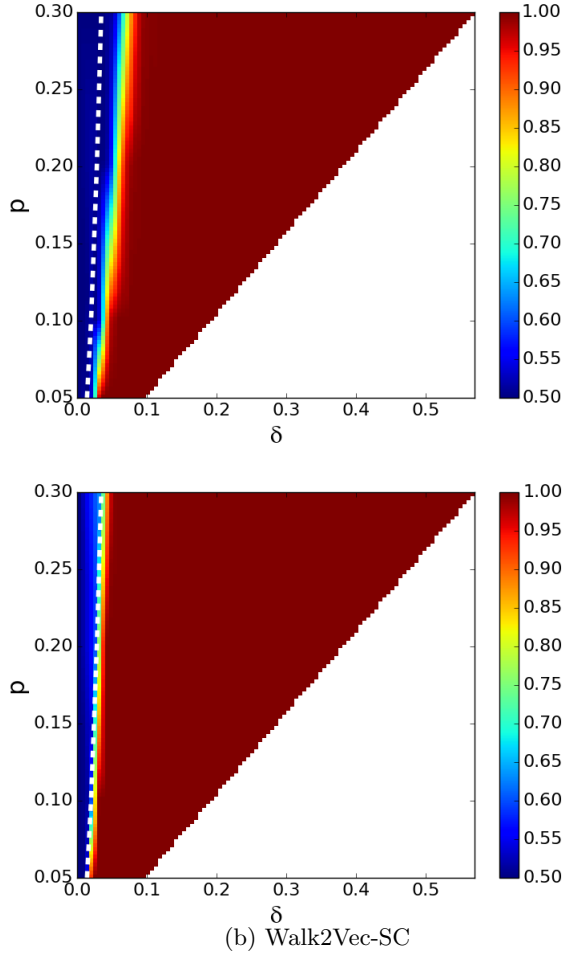


Figure 4: AUC performance on the ER vs. SBM problem. The dashed white line represents the analytical prediction of the phase transition $\delta_{\text{crit.}}$ for various densities p .

5.2 Planted Clique Problem In this section, we consider the related problem of distinguishing between an ER graph and an ER graph with a planted clique of

size k . Let $\beta = k/\sqrt{n}$ where $n = |V|$. The classification problem gets harder as the size of the clique k becomes smaller. As shown in [32], the limit of detecting a planted clique graph from a ER graph is

$$(5.10) \quad \beta_{\text{crit.}} = \sqrt{\frac{p}{1-p}},$$

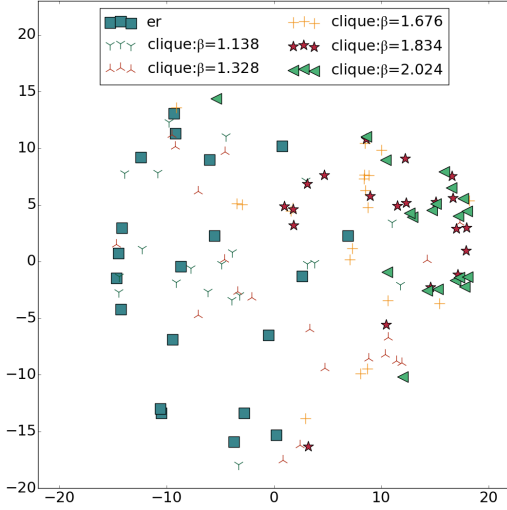
where p is the connection probability of ER graphs.

To generate a graph with a planted clique, we first generate an ER graph with density p . Then we randomly select k nodes from the graph and connect all pairs of distinct nodes in the selected node set. For each pair of (p, β) , we generate 1000 ER graphs with density p and 1000 ER graphs of the same density and with a planted clique of size k such that $k/\sqrt{n} = \beta$. All graphs are generated with $n = 1000$ number of nodes.

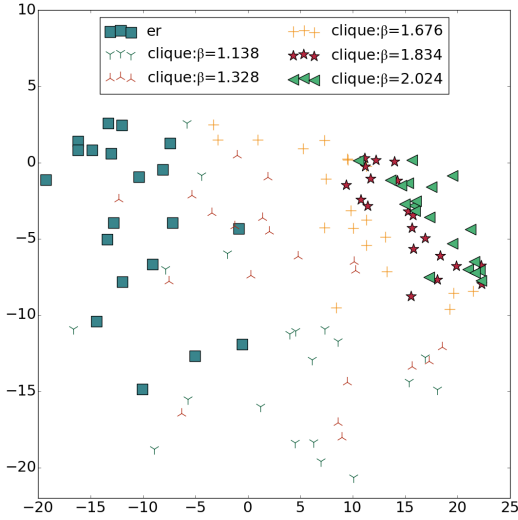
Figure 5 shows the 2-dimensional embeddings of the vector representations $\phi(G)$ of ER graphs with $p = 0.5$ and ER graphs with planted cliques. All graphs are of the size $n = 1000$. The embedding is performed using PCA. Observe from both Figure 5(a) and 5(a) that graphs with large planted clique (i.e., large β value) are further away from the ER graphs. As β decreases, these graphs move closer to the ER graphs.

After mapping each graph into its vector representation $\phi(G)$ using the proposed methods, we then train a random forest classifier on the 500 vector instances of each model and tested on the 500 vector instances of each model. Fig. 6 shows AUC performance of the planted clique problem for various graph densities p and clique parameter values β . The dark red correspond to $AUC \approx 1$ and the dark blue corresponds to $AUC \approx 0.5$. The dashed line represent the theoretical limit for clique detection. Observe from Fig. 6 that both Walk2Vec and Walk2Vec-SC systems perform well. The agreement between the theoretical limit and the simulations is excellent for $n = 1000$. As n increases, the transition is expected to get sharper.

5.3 Performance Comparison We compare the performance of the Walk2Vec and Walk2Vec-SC embeddings with that of the graph topological feature embeddings discussed in [5]. We consider the following 26 topological features: degree centrality (1-4), betweenness centrality (5-8), closeness centrality (9-12), clustering coefficient (13-16), diameter (17), radius (18), triad count (19-22), average shortest path length (23-26). Similarly to the experimental setup in [5], if one feature is assigned four numbers, this means we considered the maximum, the minimum, the average and the standard deviation over each node in the graph. We train a random forest classifier and use the learned model to classify a collection of unlabeled graph in-



(a) Walk2Vec

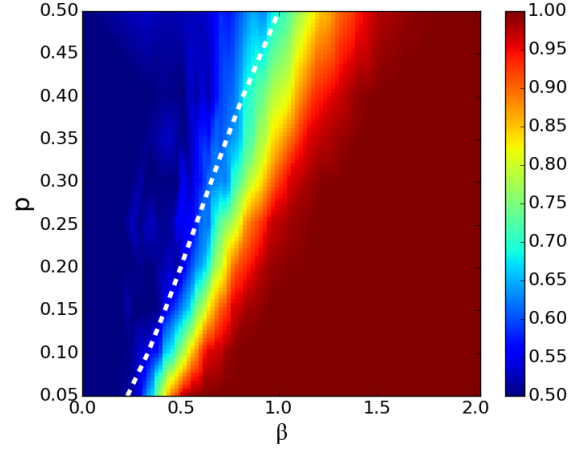


(b) Walk2Vec-SC

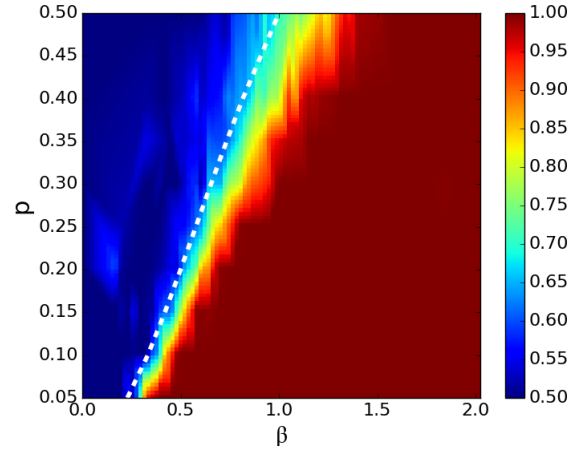
Figure 5: Two-dimensional embeddings of the vector graph representations of ER graphs and ER graphs with planted cliques.

stances.

Table 1 shows the AUC performance comparison of topological features, Walk2Vec and Walk2Vec-SC on ER graphs and SBM graphs for different δ values. All graphs are generated with $n = 1000$ number of nodes and $p = 0.05$. The horizontal dashed line in the table represents the location of the theoretical limit for discriminating ER graphs and SBM graphs.



(a) Walk2Vec



(b) Walk2Vec-SC

Figure 6: AUC performance of the planted clique problem. The dashed line represents the analytical phase transition prediction $\beta_{\text{crit.}}$.

Observe that the performance is comparable between topological features and Walk2Vec representation, while Walk2Vec representation for a graph instance is considerable cheaper than the topological features. Additionally, the Walk2Vec-SC performs the best out of the three. The phase transition is very sharp around the threshold, i.e., the dashed line.

Table 2 shows the AUC performance comparison of the three methods on ER graphs with and without planted cliques. All ER graphs are generated with $n = 1000$ and $p = 0.5$. Note that the value of β corresponds to the size of the planted clique. The horizontal dashed line represents the location of the theoretical value of the phase transition. Observe from

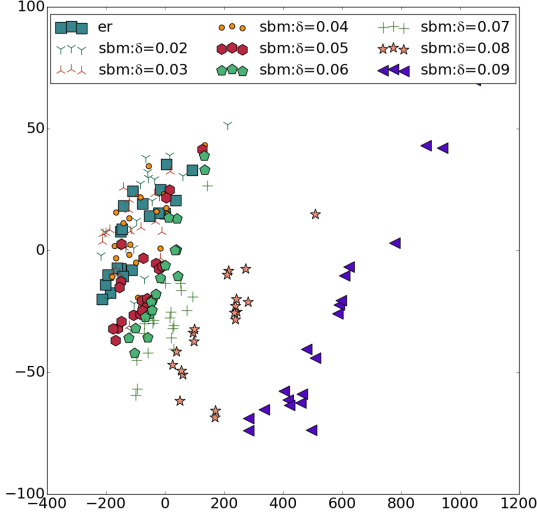


Figure 7: Two-dimensional embeddings of the topological features of ER graphs and SBM graphs

Table 1: Performance comparison on the ER vs. SBM problem. Here $n = 1000$ and $p = 0.05$.

δ	Topological Feats.	Walk2Vec	Walk2Vec-SC
0.005	0.51	0.48	0.52
0.008	0.49	0.52	0.48
0.011	0.52	0.52	0.51
0.014	0.56	0.50	0.61
0.017	0.68	0.52	0.78
0.02	0.82	0.56	0.95
0.023	0.92	0.72	0.998
0.026	0.98	0.90	1.0
0.03	0.999	0.99	1.0
0.04	1.0	0.999	1.0
0.05	1.0	1.0	1.0
0.06	1.0	1.0	1.0
0.07	1.0	1.0	1.0
0.08	1.0	1.0	1.0

Table 2 that the performance is comparable between the topological features and Walk2Vec-SC representations. Both exhibit very sharp transition around the threshold. On the other hand, the phase transition of Walk2Vec is not as sharp. This method could be useful for detecting large cliques as it is the most computationally efficient method of graph vector representation among the three methods.

A notable difference between the random walk representations generated by our two methods and the topological representation of [5] is the quality of clustering of different graph instances of the same model parameter. As illustrated in Figure 7 and

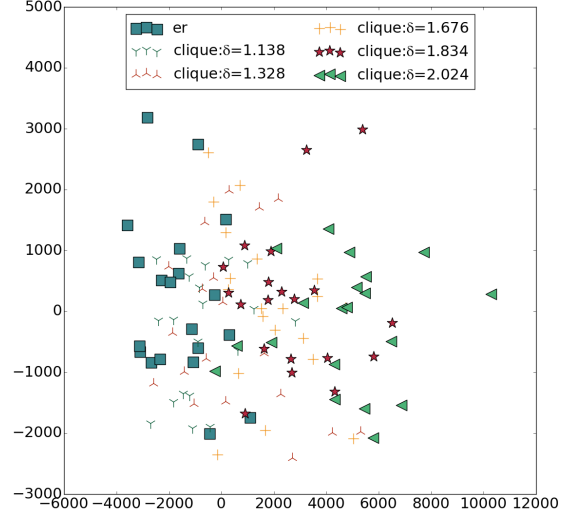


Figure 8: Two-dimensional embeddings of the topological features of ER graphs with and without cliques

Table 2: Performance comparison on the planted clique problem. Here $n = 1000$ and $p = 0.5$ for all ER graphs.

k	β	Topological Feats.	Walk2Vec	Walk2Vec-SC
10	0.316	0.48	0.51	0.50
21	0.664	0.52	0.54	0.60
31	0.980	0.71	0.62	0.71
33	1.044	0.77	0.67	0.77
36	1.138	0.87	0.70	0.84
39	1.233	0.94	0.77	0.92
42	1.328	0.98	0.81	0.97
47	1.486	0.999	0.90	0.998
53	1.676	1.0	0.97	1.0
58	1.834	1.0	0.99	1.0
64	2.024	1.0	1.0	1.0

8, topological feature embeddings appear to be much more sensitive to variations due to model randomness, and therefore, less robust in capturing the inherent structural similarity of instances generated by the same model parameter. By contrast, the Walk2Vec and Walk2Vec-SC representations appear do a much better job in smoothing out randomness effects. In realistic scenarios, we expect Walk2Vec and Walk2Vec-SC to be much more robust in handling inherent noise in observed graph instances.

In addition, both Walk2Vec and Walk2Vec-SC are scalable to large graphs. The Walk2Vec computes the random walk features on selected nodes and stack the feature vectors. For sparse graphs, the Walk2Vec's com-

putation complexity is $O(n)$ where n is the number of nodes in the graph. For Walk2Vec-SC, since the random walk feature is computed for every node in the graph and the dictionary learning is linear, the computation complexity for Walk2Vec-SC is thus $O(n^2)$. Note that one can parallelize the computation of the random walk feature for each node, thus the time it takes to compute the Walk2Vec-SC representation can be shortened significantly, making it also scalable to large graphs. On the other hand, the computation of topological features is dominated by the average shortest path length, whose computation complexity is $O(n^2 \log n)$ for directed graphs and $O(n^2)$ for undirected graphs, while Walk2Vec and Walk2Vec-SC apply to both weighted and unweighted graphs.

6 Conclusion

In this paper, we propose Walk2Vec, a novel approach that uses random walks for learning robust graph representations. Our method learns discriminating features by leveraging different mechanisms to initiate random walks and by correlating temporal dependencies between random walk steps. These representations are invariant under graph isomorphism and graph size. Experimental results on two challenging graph model selection problems show classification performance that closely matches known theoretical limits, implying that the Walk2Vec approach can map graphs into meaningful representations. Furthermore, these learned representations are robust to inherent randomness or noise in the data generation process, while simple and scalable to compute for large graphs.

References

- [1] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proc. of 20th ACM SIGKDD Inter. Conf. on Knowledge Discovery and Data Mining*, pages 701–710, 2014.
- [2] Pinar Yanardag and S.V.N. Vishwanathan. Deep graph kernels. In *Proc. of 21th ACM SIGKDD Inter. Conf. on Knowledge Discovery and Data Mining*, pages 1365–1374, 2015.
- [3] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. subgraph2vec: Learning distributed representations of rooted sub-graphs from large graphs. *CoRR*, abs/1606.08928, 2016.
- [4] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proc. of 22nd ACM SIGKDD Inter. Conf. on Knowledge Discovery and Data Mining*, pages 855–864, 2016.
- [5] Rajmonda S. Caceres, Leah Weiner, Matthew C. Schmidt, Benjamin A. Miller, and William M. Campbell. A model selection framework for graph-based data. *arXiv:1609.04859*, 2016.
- [6] Edoardo M. Airolidi, Xue Bai, and Kathleen M. Carley. Network sampling and classification: An investigation of network model representations. *Decision Support Systems*, 51(3):506 – 518, 2011.
- [7] Manuel Middendorf, Etay Ziv, Carter Adams, Jen Hom, Robin Koytcheff, Chaya Levovitz, Gregory Woods, Linda Chen, and Chris Wiggins. Discriminative topological features reveal biological network mechanisms. *BMC Bioinformatics*, 5:181, 2004.
- [8] Jeannette Janssen, Matt Hurshman, and Nauzer Kalyaniwalla. Model selection for social networks using graphlets. *Internet Mathematics*, 8(4):338–363, 2012.
- [9] Ping Zhu and Richard C. Wilson. A study of graph spectra for comparing graphs. In William F. Clocksin, Andrew W. Fitzgibbon, and Philip H. S. Torr, editors, *BMVC*. British Machine Vision Association, 2005.
- [10] Damien Fay, Hamed Haddadi, Steve Uhlig, Liam Kil-martin, Andrew W. Moore, Jrme Kunegis, and Marios Iliofotou. Discriminating graphs through spectral projections. *Computer Networks*, 55(15):3458–3468, 2011.
- [11] Daniel Yasumasa Takahashi, Joo Ricardo Sato, Carlos Eduardo Ferreira, and Andr Fujita. Discriminating different classes of biological networks by analyzing the graphs spectra distribution. *PLoS ONE*, 7(12):1–12, 12 2012.
- [12] S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. Graph kernels. *J. Mach. Learn. Res.*, 11:1201–1242, August 2010.
- [13] Francis R. Bach. Graph kernels between point clouds. In *Machine Learning, Proc. of the 25 Inter. Conf. (ICML)*, pages 25–32, 2008.
- [14] Karsten M. Borgwardt, Cheng Soon Ong, Stefan Schönerauer, S. V. N. Vishwanathan, Alex J. Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(1):47–56, 2005.
- [15] Thomas Grtner, Peter Flach, and Stefan Wrobel. On graph kernels: Hardness results and efficient alternatives. In *Conf. on Learning Theory*, pages 129–143, 2003.
- [16] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Marginalized kernels between labeled graphs. In *ICML*, volume 3, pages 321–328, 2003.
- [17] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2:1–46, 1993.
- [18] David Aldous and Jim Fill. Reversible markov chains and random walks on graphs, 2002.
- [19] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer, 2005.
- [20] James A Clarkson and C Raymond Adams. On definitions of bounded variation for functions of two variables. *Trans. of the American Mathematical Society*, 35(4):824–854, 1933.
- [21] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statis-*

- tics*, 22(1):79–86, 1951.
- [22] Don Johnson and Sinan Sinanovic. Symmetrizing the Kullback-Leibler distance. *IEEE Trans. on Information Theory*, 2001.
 - [23] Jianchao Yang, Kai Yu, Yihong Gong, and Thomas Huang. Linear spatial pyramid matching using sparse coding for image classification. In *Computer Vision and Pattern Recognition*, pages 1794–1801. IEEE, 2009.
 - [24] Youngjune L. Gwon, William M. Campbell, Douglas Sturim, and H. T. Kung. Language recognition via sparse coding. In *Proc. Interspeech*, 2016.
 - [25] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, pages 267–288, 1996.
 - [26] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.
 - [27] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.
 - [28] Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.
 - [29] P. Erdős and A. Rényi. On random graphs, I. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.
 - [30] T. A.B. Snijders and K. Nowicki. Estimation and prediction for stochastic blockmodels for graphs with latent block structure. *Journal of Classification*, 1997.
 - [31] Raj Rao Nadakuditi and Mark EJ Newman. Graph spectra and the detectability of community structure in networks. *Physical review letters*, 108(18):188701, 2012.
 - [32] Raj Rao Nadakuditi. On hard limits of eigen-analysis based planted clique detection. In *2012 IEEE Statistical Signal Processing Workshop*, pages 129–132, 2012.